REMARKS

Claims 1-28 are pending in the present application, were examined and stand rejected. In response, Claims 1, 3, 4, 8, 10-12, 15, 18, 19, 22, 24 and 28 are amended, no claims are cancelled and no claims are added. Applicant respectfully requests reconsideration of pending Claims 1-28 in view of at least the following remarks. Reconsideration and withdrawal of the rejections of record are requested in view of such amendments and the following discussion.

I. Claims Rejected Under 35 U.S.C. §112

The Examiner has rejected Claims 1-28 under 35 U.S.C. §112, first paragraph, as failing to comply with the written description requirement. Applicant respectfully traverses this rejection.

Applicant respectfully submit that the original disclosure establishes that he/she was in possession of the amended subject matter. For at least for the reasons provided below, Applicant respectfully submit that the original disclosure of the application as filed inherently includes the amended feature, previously added to claims 1, 8, 15, and 22, to recite that identification of unifiable variables is performed prior to register allocation.

Applicants respectfully submits that identification of unifiable variables must necessarily be performed prior to register allocation, since such variable would necessarily be lost once register assignment is performed. As indicated by Applicant's specification, unifiable variables that are not simultaneously used in the executable instruction are identified with each subgraph structure. (See page 4, paragraph 15, lines 4-5 of Applicant's specification. As further indicated by Applicant's specification, unifiable instructions contain one or more unifiable variables.

Regarding the requirement that unifiable variables are not simultaneously used, as further described for example with reference to FIG. 3 of Applicant's specification, an interference graph structure is provided to indicate which variables of the local variables are simultaneously used in the executable instructions within the times and cannot be unified since such variables have overlapping live ranges. (See page 7, paragraph 22 of Applicant's specification) Accordingly, the requirement to identify variables which do not have overlapping live ranges must be performed prior to register assignments.

Further Applicants have amended the specification to illustrate that figure 1 illustrates source code (specifically C source code) and not executable code, as indicated by Applicant's

specification. (See Exhibit A for the attached Webopedia definition of "source code.") As known to those skilled in the art, source code is the first to code generated by a programmer using a high level programmer language and does refer to register assignment of variables such as used in for example assembly language code. (See Exhibit B for Webopedia definition of "assembly language.")

As known to those skilled in the art most architectures have a limited number of registers. As a result, such registers may be re-used which makes it very difficult to track variables once register allocation has been performed. Accordingly, Applicants respectfully submit that the identification of unifiable variables from source code inherently requires that such identification of unifiable variables is performed from source code of program and not from assembly language or executable of a program which includes register assignment for variables.

Accordingly, for at least the reasons provided above, Applicant respectfully submits that the original disclosure establishes that Applicant was in possession of the amended subject matter at the time of filing of the patent application. Consequently, Applicants respectfully submit that claims 1-28 is amended contain subject matter which was described in the specification in such a way as to reasonably convey to one skilled in the relevant art that the Applicant comment the time the application was filed, have possession of the claimed invention as required to comply with 35 U.S.C. § 112 first paragraph.

Therefore, for the reasons provided above Applicant respectfully request that the Examiner reconsider and withdraw this section that 35 U.S.C. § 112 first paragraph rejection of claims 1-28.

II. Claims Rejected Under 35 U.S.C. §103

The Examiner has rejected Claims 1-28 under 35 U.S.C. §103(a) as being unpatentable over "Enhanced Code Compression for Embedded RISC Processors," by Cooper et al. ("Cooper") in view of U.S. Patent No. 6,918,111 issued to Damron et al. ("Damron"). Applicant respectfully traverses this rejection.

Regarding Claims 1 and 15, Claims 1 and 15 are amended to recite the following claim features which are neither taught nor suggested by the prior art combination of Cooper in view of Damron:

unifying each unifiable variable within said at least one unifiable instruction.

As indicated by the above recited feature of amended Claims 1 and 15, Claims 1 and 15 are amended to recite features of dependent claims 5 and 19, respectfully. According to the Examiner, features of Claims 5 and 19, which are added to amended claims 1 and 15 respectively, are taught by Cooper at page 40 left col. 8-9 and page 141 right col. Lines 27-29. (See page 7, paragraph 2 of the Office Action mailed 5/5/06.) As indicated by the first passage cited by the Examiner:

> Sections 2 and 3 describe how our compression framework identifies "repeats, then replaces fragments within a repeat with references to a single shared instance." (page 140 left col. Lines 5-8.")

Regarding the term "repeats", Cooper defines the term "repeat" as follows: "pattern-matching techniques identify identical code sequences (a repeat.)" (page 139, right col., lines 20-21.) Applicants respectfully submit that the identification of identical code sequences or repeats as taught by Cooper fails to teach or suggest the unification of variables of at least one unifiable instruction as recited by amended Claims 1 and 15.

According to the other passage cited by the Examiner:

> Figure 4 shows an example of cross-jumping (sometimes known as tail merging), in which identical regions that end with a jump to the same target are merged together. [23] In this transformation, we replace a region with a direct jump to another identical region all the out branches in the region mismatch in order for cross/jumping to be applied. (page 141, right col. Lines 26-32.)

Applicant respectfully submits that the above cited passage incorporates a jump instruction in order to share common (repeated) code sections. Applicant respectfully submits that the above recited passage provides no disclosure, teaching or suggestion regarding to unification of variables of at least one unifiable instruction, as recited by amended Claims 1 and 15.

As disclosed by Cooper, Cooper uses either procedural abstraction, as shown in FIG. 3 of Cooper, or cross jumping, as shown in FIG. 4, to channel execution of repeats through a single copy of the code. Cooper teaches the expansion of the basic algorithm of using a pattern matching technique to identify "identical" code sections (repeats) by relaxing the notion of "identical" to abstract the away register names, which Cooper indicates is a key enhancement when compressing code compiled with a graph-coloring register allocator. (See page 139 right col. Lines 16-27.) Applicants respectfully submit that cross jumping as shown in FIG. 4, as well as procedural abstraction, as shown in FIG. 3, where a given code region is made into a procedure, and other identical regions to it are replaced with calls to the new procedure (See page 141, right col. Lines 18-22,) neither teache nor suggest unification of variables, as recited by amended Claims 1 and 15.

As amended by case law, "to establish *prima facie* obviousness of a claimed invention, all claimed imitations must be taught or suggested by the prior art." In re, Royka 490 f.2d 981, 180 USPQ 580 (CCPA 1974)

Here, the disclosure and Cooper is directed to extending a pattern matching technique for identifying identical code sequences, which are referred to as "repeats," by using procedure abstraction or cross-jumping to channel execution of "repeats" to a single copy of the code. These techniques are expanded by Cooper by relaxing the notion of "identical" to abstract the way register names. (See Cooper, page 139, right col. Lines 20-27.)

Applicants respectfully submits that the identifying of identical code sections where the register names are abstracted away, does not provide any teachings or suggestions with regards to the unifying of variables of at least one unifiable instruction as recited by amended Claims 1 and 15. In other words, the identifying of identical code sections, where the notion of identical is

relaxed to abstract the way register names, as taught by Cooper, fails to teach of suggest the unification of variables as recited by amended Claims 1 and 15.

Regarding the Examiner's citing of Damron, the Examiner's citing of Damron fails to rectify the deficiencies of Cooper in failing to teach or suggest unifying each unifiable variable within said at least one unifiable instruction, as recited by amended Claims 1 and 15. Consequently, Applicants respectfully submit that Applicant's amendments to Claims 1 and 15 prevent the Examiner from establishing *prima facie* case of obviousness of amended Claims 1 and 15 since all claim features recited by amended Claims 1 and 15 are neither taught or suggested by the prior art combination in view of Cooper in view of Damron, as required to establish a *prima facie* case of obviousness. Id.

Accordingly, for the reasons provided above, Applicants respectfully submit that Claims 1 and 15 as amended are patentable over the combination in view of Cooper in view of Damron as well as the references of record. Therefore, Applicants respectfully request that the Examiner reconsider and withdraw the Section 103 rejection of Claims 1 and 15.

Regarding Claims 2-14 and 16-21 Claims 2-14 and 16-21 base their dependency from amended claims 1 and 15, respectively are also patentable over the combination of Cooper in view of Damron. Consequently, Applicant respectfully request that the Examiner reconsider and withdraw the section 103 rejection of Claims 2-14 and 16-21.

Regarding Claims 8 and 12, claims 8 and 12 recite the following analogous claim feature which is neither taught nor suggested by the prior art combination of Cooper in view of Damron:

unifying each of unifiable variable within at least one unifiable instruction.

Applicants respectfully submit that the above recited features of amended Claims 8 and 22 are analogous to the above recited features of Claims 1 and 15. Accordingly, Applicant's

arguments provided above with regard to the section 103(a) rejection of amended claims 1 and 15 is obvious over Cooper in view of Damron, equally apply to the Examiner's section 103(a) of Claims 8 and 22 is obvious over the indicated prior art.

Consequently, at least for the reasons provided above, Applicants respectfully submit the amended Claims 8 and 22 are patentable over the combination of Cooper in view of Damron since the combination fails to teach or suggest each of the above recited features of amended Claims 8 and 22 and specifically the unifying of each unifiable variable within a set of at least of one unifiable instruction as recited by amended claims 8 and 22. Therefore, Applicants respectfully request that the Examiner reconsider and withdraw the section 103(a) rejection of Claims 8 and 22.

Regarding Claims 9-14 and 23-28, Claims 9-14 and 23-28 based on their dependency from Claims 8 and 22, respectively, are also patentable over the combination of Cooper in view of Damron. Consequently, Applicants respectfully request that the Examiner reconsider and withdraw the section 103(a) rejection of Claims 9-14 and 23-28.

CONCLUSION

In view of the foregoing, it is submitted that Claims 1-28, as amended, patentably define the subject invention over the cited references of record, and are in condition for allowance and such action is earnestly solicited at the earliest possible date. If the Examiner believes a telephone conference would be useful in moving the case forward, he is encouraged to contact the undersigned at (310) 207-3800.

If necessary, the Commissioner is hereby authorized in this, concurrent and future replies, to charge payment or credit any overpayment to Deposit Account No. 02-2666 for any additional fees required under 37 C.F.R. §§1.16 or 1.17, particularly, extension of time fees.

Respectfully submitted,

BLAKELY, SOKOLOFF, TAYLOR, & ZAFMAN LLP

Dated: August 2, 2006

By: _____
Joseph Lutz, Reg. No. 43,765

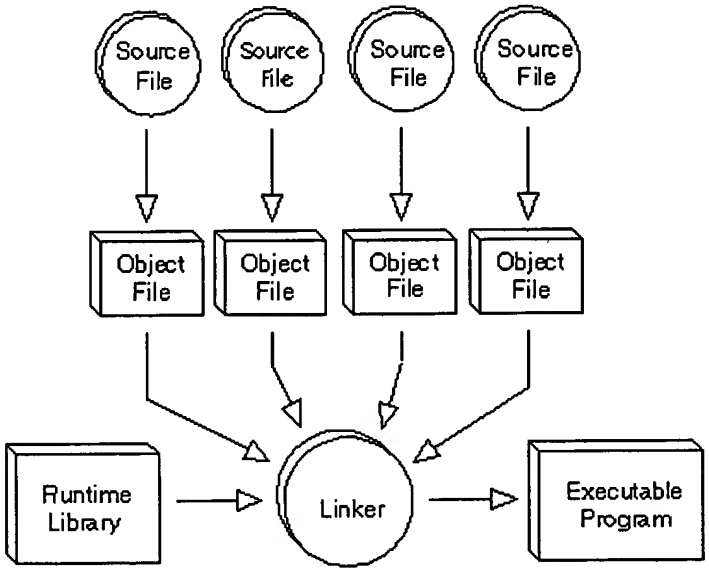
12400 Wilshire Boulevard
Seventh Floor
Los Angeles, California 90025
(310) 207-3800

CERTIFICATE OF FACSIMILE:
I hereby certify that this correspondence is being transmitted via facsimile on the date shown below to the United States Patent and Trademark Office.

_____
Melissa Stead August 2 , 2006

source code

Last modified: Sunday, September 01, 1996



Program instructions in their original form. The word *source* differentiates code from various other forms that it can have (for example, object code and *executable code*).

Initially, a programmer writes a program in a particular programming language. This form of the program is called the *source program,* or more generically, *source code*. To execute the program, however, the programmer must translate it into machine language, the language that the computer understands. The first step of this translation process is usually performed by a utility called a compiler. The compiler translates the source code into a form called object code. Sometimes the object code is the same as machine code; sometimes it needs to be translated into machine language by a utility called an assembler.

Source code is the only format that is readable by humans. When you purchase programs, you usually receive them in their machine-language format. This means that you can execute them directly, but you cannot read or modify them. Some software manufacturers provide source code, but this is useful only if you are an experienced programmer.

•E-mail this definition to a colleague•

Related Categories

Compiling, Binding and

*For internet.com pages about **source code CLICK HERE**. Also check out the following links!*

LINKS = Great Page!

Sponsored listings

Linking

Related Terms

assembler

break-even point

build

code

compiler

CVS

editor

fork

KLOC

machine language

object code

program

programming language

refactoring

SOA

software entropy

(Webopedia)

Give Us Your Feedback

assembly language

Last modified: Wednesday, May 02, 2001

MENU

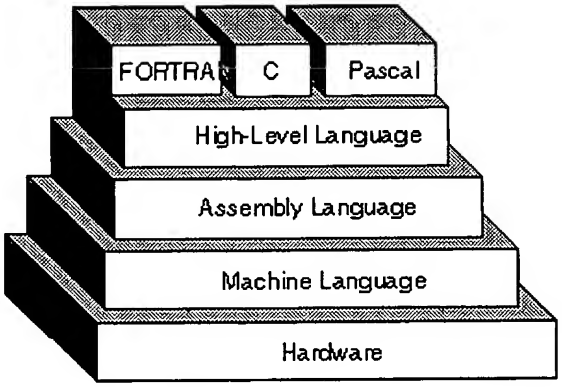
Home
Term of the Day
New Terms
Pronunciation
New Links
Quick Reference
Did You Know?
Categories
Tech Support
Webopedia Jobs
About Us
Link to Us
Advertising

Compare Prices: go

HardwareCentral

Talk To Us...

Submit a URL
Suggest a Term
Report an Error



A programming language that is once removed from a computer's machine language. Machine languages consist entirely of numbers and are almost impossible for humans to read and write. Assembly languages have the same structure and set of commands as machine languages,

internet.com
Developer
International
Internet Lists
Internet News
Internet Resources
IT
Linux/Open Source
Personal Technology
Small Business
Windows Technology
xSP Resources
Search internet.com
Advertise
Corporate Info
Newsletters
Tech Jobs
E-mail Offers

internet commerce
Be a Commerce Partner
Marketing Products
Web Hosting Providers
Cell Phone Plans
T-Shirts
Merchant Accounts
KVM Switches Online
Compare Prices
CRM Software
Phone Systems
Cheap Plane Tickets
Cheap Airline Tickets
Online Universities
Online Masters
Domain Registration

but they enable a programmer to use names instead of numbers.

Each type of CPU has its own machine language and assembly language, so an assembly language program written for one type of CPU won't run on another. In the early days of programming, all programs were written in assembly language. Now, most programs are written in a high-level language such as FORTRAN or C. Programmers still use assembly language when speed is essential or when they need to perform an operation that isn't possible in a high-level language.

•E-mail this definition to a colleague•

Related Categories

*For internet.com pages about **assembly language CLICK HERE**. Also check out the following links!*

LINKS

= Great Page!

Assembly Internet resources

Contains a collection of resources for assembly language including books, FAQs, coding resources, tutorials, newsgroups, and hot lists.

Assembly Language Resources

Comprehensive collection of resources that includes tutorials, FAQs, related sites, vendor lists, and conference information.

Assembly programming resources

Provides a comprehensive set of links to assembly programming topics. The information is organized in a table and offers the following categories: assembly world wide Web pages, newsgroups, FAQs, FTP sites, and technical documentation.

Yahoo!'s assembly language page

Yahoo!'s directory of assembly language links.

Programming Languages

Related Terms

assembler

KLOC

(Webopedia)

Give Us Your Feedback